Algorithme et Langage C

Les structures de contrôle ~ Itération

Fiche n° 2

Définition:

Une itération représente le bouclage sur une suite d'instructions afin d'éviter une redondance de code. Ces itérations sont soumises à l'évaluation d'un prédicat pour ne pas boucler indéfiniment. Ce prédicat constitue soit une condition de maintien, soit une condition de sortie de la boucle suivant le cas.

Boucle indéfinie:

Ce type d'itération est utilisé lorsque le nombre de répétitions n'est pas déterminé avant le début de la boucle. Il est cependant nécessaire que la valeur du prédicat puisse évoluer à l'intérieur de cette boucle sous peine de rester bloquer à l'intérieur. L'évaluation du prédicat peut être positionnée soit avant la suite d'instructions, soit après. Le premier cas conditionne l'entrée dans la boucle, le second impose de réaliser au moins une fois la suite d'instructions

Évaluation du prédicat avant la boucle :

L'évaluation du prédicat représente la condition de maintien Le traitement est réalisé tant que la valeur du prédicat est vraie

Algorithmique

Tant que prédicat

Traitement

FinTantQue

```
Langage C
while ( /* prédicat */ )
{
    /* Traitement */
}
```

Le prédicat est toujours exprimé entre parenthèses en langage C. Il n'y a pas de point-virgule à la suite.

Le bloc d'instructions est défini entre accolades, il représente le traitement à effectuer tant que le prédicat est VRAI.

Si le traitement comporte une seule instruction, les accolades ne sont pas obligatoires.

Évaluation du prédicat avant la boucle :

Répéter

Traitement

Jusqu'à prédicat

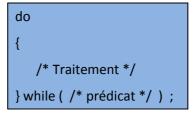
Le prédicat exprime la condition de sortie de la boucle <u>Faire</u>

Traitement

<u>Tant que</u> prédicat

Le prédicat exprime la condition de maintien dans la boucle

Attention pour passer de l'un à l'autre le prédicat doit être inversé.



Pour le langage C, seule la forme avec l'évaluation du prédicat sur la condition de maintien est conservée.

Le **do**... **while** est terminé par un point-virgule après le prédicat.

Le bloc d'instructions est défini entre accolades, il représente le traitement à effectuer tant que le prédicat est VRAI.

Boucle définie:

Ce type d'itération est utilisé lorsque le nombre de répétitions est connu avant le début de la boucle. Un indice sert à comptabiliser le nombre d'itérations effectué. Il est initialisé à une valeur de départ, puis il est évalué afin de vérifier si cette valeur permet de faire une première fois le traitement. Après la dernière instruction constituant le corps de cette boucle, il est ensuite incrémenté pour passer à la valeur suivante. La condition de maintien est alors de nouveau évaluée pour savoir si un autre passage doit être effectué, ainsi de suite à chaque passage.

Initialisation de l'indice à 0 Condition de l'indice de l'indice de l'indice de l'indice de l'indice for (indice = 0 ; indice < = NB_VAL ; indice ++) {

/* traitement */

<u>Pour</u> indice <u>allant de</u> 0 <u>à</u> NB_VAL <u>pas de</u> 1 traitement

FinPour

L'indice est la variable permettant de compter le nombre de tours de boucle effectué. Ici, ce nombre sera NB_VAL + 1 comme l'indice commence à 0.

Le pas de l'indice est facultatif lorsqu'il est égal à 1. NB_VAL représente ici une constante. Des variables peuvent être utilisées pour valeurs initiale et finale. L'incrémentation de l'indice n'est pas représentée, il est implicite.

Ce type d'itération est largement utilisé avec les tableaux, l'indice de la boucle servant souvent d'indice pour parcourir les cases du tableau. La parenthèse du **for** est composée de 3 parties séparées par des points-virgules, la première initialise la variable, la seconde teste la condition de maintien, c'est l'évaluation d'un prédicat, la troisième représente le moyen de passer à la valeur suivante

La parenthèse fermante n'est pas suivie de point-virgule, sinon le traitement n'est pas effectué.

```
int indice = 0;
while (indice <= NB_VAL)
{
    /* traitement */
    indice++;
}</pre>
```

Cette portion de code est équivalant à la boucle *for* présentée ci-dessus.